HW5
10.2
a)
See python code for implementation
A1:0
A2:2
A3:1
B1:1
B2:1
B3:1
C1:2
C2:1

b)
The final result is: See code for process.
A1:0
A2:2
A3:1
B1:0
B2:1
B3:1
C1:2
C2:0

10.4
k-means ++ is effective because when choosing the initial point of k-cluster, it respects that probability of the choice of a center should be proportional to the squared-distance to a given point. This guarantees that centers tend to be far away from each other. Given centers are more separate, k-means should converge faster, because the initial condition are relatively closer to the objective—separating points into different groups. Also the result should be better, because initial points are further away, and thus prevent bad local convergence where points are fuzzy together.

10.6
a)
k-medoids are more robust against noises, because its ways of calculating the medoid is less influenced than simply choosing the mean. However it is more computationally expensive than k-means.

b)k-means and k-medoids are partition based algorithm and computationally cheap and allows the possibility of major changes of the shape of cluster, because in each iteration every point is re-clustered. However hierarchical-based method suffers from the disadvantage that once a split or merge happen it cannot reverse. Usually partition-based is better off hierarchical-based method. However the prerequisite of knowing the number of partition could be a drawback.

9.1

SVM:

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     senior
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     junior
Weighted Avg.    1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000

=== Confusion Matrix ===

   a   b   <-- classified as
  52   0 |   a = senior
   0 113 |   b = junior
```

Test option is selected as 66-34 percent split, and the confusion table show above, the accuracy is 100%

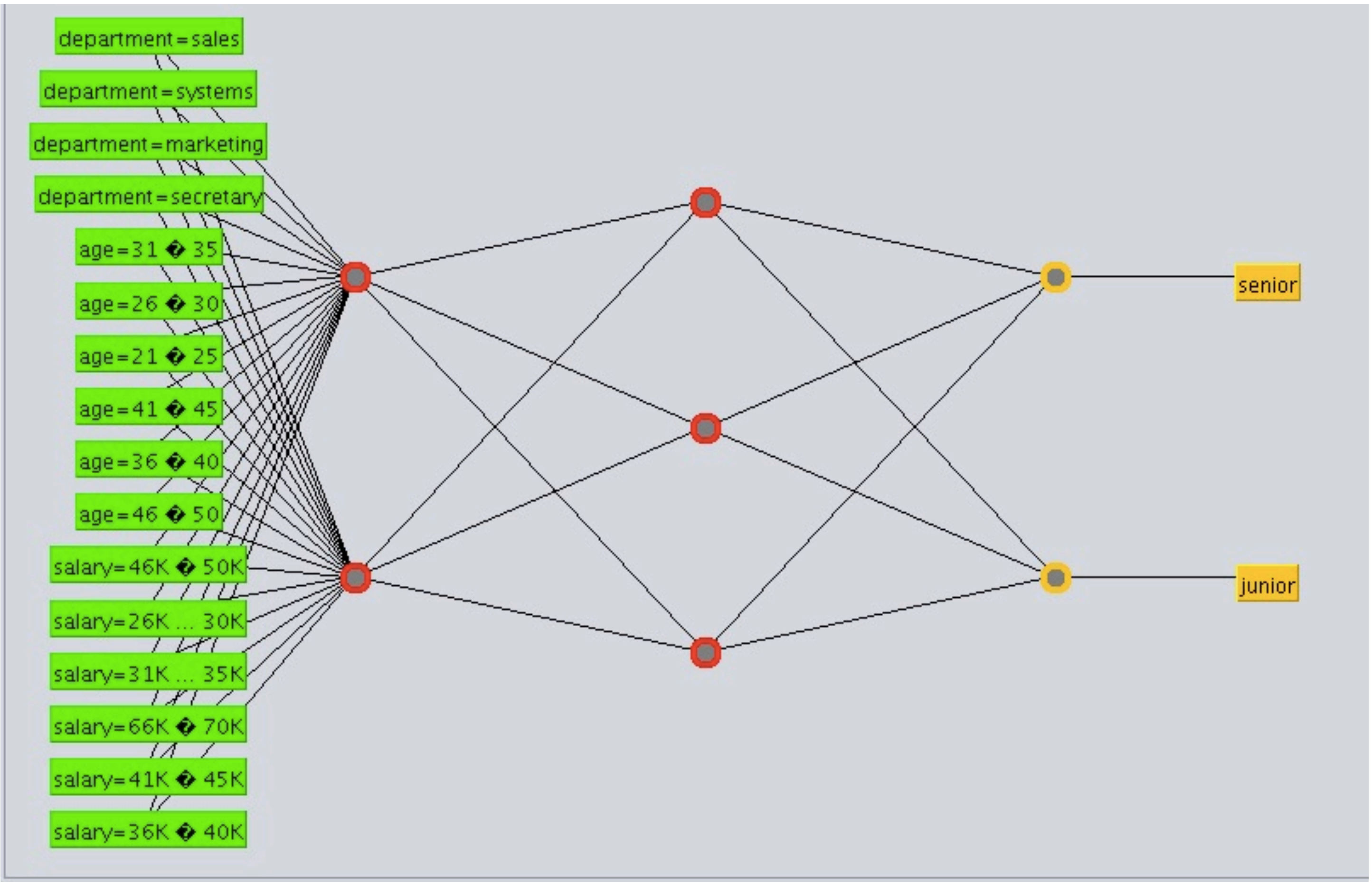
Navie Bayes:

```
   === Summary ===

   Correctly Classified Instances          56              100      %
   Incorrectly Classified Instances         0                0      %
   Kappa statistic                          1
   Mean absolute error                      0.1514
   Root mean squared error                  0.2271
   Relative absolute error                 33.3442 %
   Root relative squared error             45.4192 %
   Total Number of Instances               56

   === Detailed Accuracy By Class ===

                    TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                    1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     senior
                    1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     junior
   Weighted Avg.    1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000

   === Confusion Matrix ===

     a  b   <-- classified as
    22  0 |  a = senior
     0 34 |  b = junior
```

The result shows that naïve Bayes gives also 100% accuracy.

MLP:

Input and out put are shown as the image below:



where weight and bias are shown as below:

a)

```
=== Run information ===

Scheme:      weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H "2, 3" -G -R
Relation:    9.1  data
Instances:    165
Attributes:   4
              department
              status
              age
              salary
Test mode:    split 66.0% train, remainder test

=== Classifier model (full training set) ===

Sigmoid Node 0
    Inputs    Weights
    Threshold    -0.04479154618260167
    Node 4    0.04731076511749595
    Node 5    -0.04728309381311135
    Node 6    0.013028927595620451
Sigmoid Node 1
```

```
    Inputs    Weights
    Threshold    -0.04898245454637708
    Node 4    0.011363760252725875
    Node 5    0.018901751454134197
    Node 6    0.03178375060850522
Sigmoid Node 2
    Inputs    Weights
    Threshold    -0.01461470302912881
    Attrib department=sales    -0.0455509379843205
    Attrib department=systems    -3.196236377343911E-4
    Attrib department=marketing    0.002068881989294946
    Attrib department=secretary    -0.0022363088243078644
    Attrib age=31 � 35    -0.017723220175677866
    Attrib age=26 � 30    0.02010389381824046
    Attrib age=21 � 25    0.010728824856261775
    Attrib age=41 � 45    -0.011095681644637145
    Attrib age=36 � 40    -0.03326109114155931
    Attrib age=46 � 50    -0.023510417254752836
    Attrib salary=46K � 50K    -0.020243173508921852
    Attrib salary=26K ... 30K    0.018755163818386378
    Attrib salary=31K ... 35K    -0.014694750055678031
    Attrib salary=66K � 70K    0.024041718016938274
    Attrib salary=41K � 45K    0.007995778326866723
    Attrib salary=36K � 40K    -0.018803026306620854
Sigmoid Node 3
    Inputs    Weights
    Threshold    -0.04132621448342725
    Attrib department=sales    0.01467239010388896
    Attrib department=systems    6.70159595285269E-4
    Attrib department=marketing    -0.0133639255486004
    Attrib department=secretary    0.020396970534263453
    Attrib age=31 � 35    -0.04883451617234522
    Attrib age=26 � 30    0.024535286039155085
    Attrib age=21 � 25    -0.026491827413583392
    Attrib age=41 � 45    0.04114683618429858
    Attrib age=36 � 40    -0.04960864693827791
    Attrib age=46 � 50    0.039281695715618514
    Attrib salary=46K � 50K    -0.007109552067026251
    Attrib salary=26K ... 30K    -0.009150356529897409
    Attrib salary=31K ... 35K    0.017175856749953686
    Attrib salary=66K � 70K    0.039443522450634316
    Attrib salary=41K � 45K    0.0010755479948259747
    Attrib salary=36K � 40K    0.004022940417141983
Sigmoid Node 4
    Inputs    Weights
    Threshold    0.035031151526430576
    Node 2    -0.04963230976442996
    Node 3    -0.019210683236552728
Sigmoid Node 5
    Inputs    Weights
    Threshold    0.003160855624879763
    Node 2    0.04188142018385732
    Node 3    -0.022278997393128865
Sigmoid Node 6
    Inputs    Weights
    Threshold    0.03742622102831944
    Node 2    0.010988151351276357
    Node 3    0.04086392096967359
Class senior
    Input
    Node 0
Class junior
    Input
    Node 1
```

b)

The last instance is append to the end of the whole data, in this case , the new dnn can be interpreted as fit the given instance once based on the initial condition from a). By comparing the weight change from a) to b) we can answer the question.

The learning rate is set to be 0.3.

The following weight is obtained by running back prorogation on the given instance.

```
Sigmoid Node 0
    Inputs    Weights
    Threshold    -0.031117935463524673
    Node 4    -0.04799401966685414
    Node 5    0.03779666973504053
    Node 6    -0.019919659468773766
Sigmoid Node 1
    Inputs    Weights
    Threshold    -0.0305390009971664
    Node 4    -0.008671213645981182
    Node 5    0.01769729499561977
    Node 6    0.046996093900678076
```

The node1 and node2 weight are different than the previous version. We are confident that the weight changes.

11.2
After computation, the two distance yield exactly the same distance, 9.8474368042e-07. See python code for implementation.

The reason behind this is that, according to the theorem that defines a valid distance measure, no matter what distance measure is used, the probability of a distance comparison event happen remains the same. Further illustration is that a distance comparison event contains exactly the same set basic events regardless of the choice of distance measurement.